

**IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF NEW YORK**

**INTERNATIONAL BUSINESS
MACHINES CORPORATION,**

Plaintiff,

-vs.-

**PLATFORM SOLUTIONS, INC., and
T3 TECHNOLOGIES, INC.,**

Defendants.

**DECLARATION OF DR. MARK
SMOTHERMAN IN SUPPORT OF
IBM'S OPENING CLAIM
CONSTRUCTION BRIEF**

Civil Action No. 06 CV 13565 (LAK)

DECLARATION OF DR. MARK SMOTHERMAN

I, Dr. Mark Smotherman, declare as follows:

1. I have been an Associate Professor at the Clemson University Department of Computer Science, now known as the School of Computing, since 1990. Additionally, I have been the Graduate Program Coordinator at the Clemson University School of Computing since 2004; a position I also held from 1991 to 1992. In 1977, I received a Bachelor of Science degree *summa cum laude* in Physics from Middle Tennessee State University. In 1984, I received my Ph.D. in Computer Science from the University of North Carolina at Chapel Hill. From 1984 to 1990, I was an Assistant Professor at Clemson University. I have supervised more than 30 graduate students for their M.S. degrees. I have published more than 40 scientific papers and book chapters, and serve on multiple professional committees. I have also had industrial experience as both a programmer and an analyst. Additionally, I have extensive experience as both a consultant and an expert witness in the fields of computer architecture, software and computer processors.

2. I attach a copy of my curriculum vitae, which describes in more detail my background and qualifications, as Exhibit 1 to this declaration.

3. I submit this declaration in support of Plaintiff IBM's Opening Claim Construction Brief for the *Markman* Hearing to address the interpretation and meaning of certain terms used in the claims of United States Patent Nos. 5,987,495; 6,775,789; 5,953,520; 6,009,261; 5,687,106; 5,696,709; 5,825,678; 6,654,812; 6,971,002; and 5,414,851 (collectively the "patents-in-suit"), as well as provide expert testimony regarding the level of knowledge of a person of ordinary skill in the art at the time of the inventions described in those patents.

4. In connection with the preparation of this declaration, I have reviewed the patents-in-suit, the prosecution file histories for the patents-in-suit, prior art cited during prosecution, and other references cited in this declaration. I have also considered the level of one of ordinary skill in the art, and I have concluded that it consists of a degree in electrical engineering or computer science, along with at least 2-4 years of experience related to the area of technology of the particular patent at issue.

Processor

5. For a computer system to function, it must have a processor that interprets and executes instructions. The processor interprets an instruction by examining and analyzing the operation and operands specified by that instruction, and then executes the instruction by carrying out the operation set forth by the instruction. A person of ordinary skill in the art at the time of the inventions for the asserted patents would understand that the term "processor" could include more than just "one or more integrated circuits." Although a processor could consist of one or more integrated circuits, it was also well known at the time of the inventions (and is presently known) that processors could consist of a combination of both hardware and forms of software not limited to integrated circuits.

6. "Microcode" is one example of software that is combined with hardware to form a processor. Microcode performs very simple operations as part of the interpretation and execution of instructions. Microcode works in conjunction with the integrated circuit(s) to complete the implementation of a instruction execution. The microcode portion of the processor could be contained within the integrated circuit(s) or could be separate from those integrated circuits. For example, the microcode portion of the processor can be stored on ROM or RAM separate and apart from the integrated circuit(s).

7. There were several well-known examples of processors that consisted of both hardware and microcode (software) by the time the patents were filed. The IBM System/360 computer family, first announced in 1964, is a well-known example of the use of microcode in processors. Four of the initial IBM System/360 models used microcode stored separately from the integrated circuit(s): Model 30 used a form of mylar card separate from the integrated circuits to hold the microcode; Model 40 used a form of magnetic cores separate from the integrated circuits to hold the microcode; Model 50 and 65 used capacitors separate from the integrated circuits to hold the microcode. *See* P. Fagg, J. Brown, J. Hipp, D. Doody, J. Fairclough, and J. Green, *IBM System 360 engineering*, Proc. AFIPS Fall Joint Computer Conf. (FJCC), Pt. I, vol. 26, 1964, at 205-231; E. Pugh, L. Johnson, and J. Palmer, *IBM's 360 and Early 370 Systems*, Cambridge, MA, MIT Press 1991. Similarly, the IBM System/370 Model 145 used a part of main memory RAM separate from the integrated circuits to hold its microcode. *See* IBM SYSTEM/370 MODEL 145 FUNCTIONAL CHARACTERISTICS, IBM FORM NUMBER GA24-3557 (Oct. 1970), available at http://bitsavers.vt100.net/pdf/ibm/370/funcChar/GA24-3557-1_370-145_funcChar_Oct70.pdf.

8. "Millicode" is another example of software that is combined with hardware to form a processor. Millicode consists of special software routines used to interpret and execute the more complex instructions. The millicode works in conjunction with the integrated circuit(s) to complete the implementation of a instruction execution. The millicode is typically stored in a special part of main memory separate and apart from the integrated circuit(s). The concept of using special software routines to implement complex instructions emerged at least as early as 1971. *See* M. Faix, and C. Schuenemann, *Combined Macro/Micro Program Machine*, IBM Technical Disclosure Bulletin, vol. 14, No. 1, June 1971, at 298.

Instructions and Instruction Sets

9. Computers require instructions to perform tasks. Instructions indicate to the computer system the operations to be performed, and the operands upon which those operations are performed. For example, a simple instruction for a computer may be to "ADD X to Y" – the operation is to perform the ADD function and the operands are X and Y. The term "instruction" was well known throughout the art at the time of the inventions, as it describes one of the most fundamental concepts of computing. A computer architecture is a collection of instructions, called the "instruction set," along with a definition of how operand values are interpreted.

10. Instructions differ to the extent there are different "levels" of instructions. Depending on the level of the instruction, it may require translation or interpretation before the instruction can be executed by the processor. For example, an instruction may be written at the assembly language level or the machine language level. An instruction written at the assembly language level ("an assembly instruction") would require translation into the corresponding machine instruction(s) at the machine language level before it (they) can be executed by a processor.

11. At the assembly language level, the instructions are symbolic (e.g., "ADD X, Y") and include alphanumeric names and acronyms (e.g., "X" for a variable name). At the machine language level, the instructions are typically represented by bit patterns (e.g., 10010101). Both assembly language level instructions and machine language level instructions could appropriately be called "language constructs" because both are constructed or built as a collection of elements in the vocabulary at that language level.

12. At the time the patents were filed, one of ordinary skill in the art would not have read the generic term "instruction" to mean that it must be of a particular format or the manner

by which it is executed. To the contrary, although the more specific term "machine instruction" would have normally meant that it could be directly executed by a processor, the generic term "instruction" is not limited in that way.

Program Status Word

13. A program status word is a collection of data (e.g., 64-bits or 128-bits) that represents a subset of the current values of a program during execution. This includes the address of the next instruction to be executed, the condition code, and the access "key" to different areas of main memory (i.e., the program authority).

14. A person of ordinary skill in the art at the time the '678 and '495 patents were filed in 1995 and 1997, respectively, would have understood that a program status word could be stored in a register or some other type of storage location, such as RAM. For example, the IBM System/360 Model 40 stored the program status word in a magnetic core array known as local storage. See P. Fagg, J. Brown, J. Hipp, D. Doody, J. Fairclough, and J. Green, *IBM System 360 engineering*, Proc. AFIPS Fall Joint Computer Conf. (FJCC), Pt. I, vol. 26, 1964, at 218.

Registers

15. In the context of computer architecture, a person of ordinary skill in the art at the time of the inventions would have understood the meaning of the term "register" to be the location of an operand in a restricted set of storage locations. These storage locations could have been in fast integrated circuits, a special purpose RAM, or even in main memory. For example, the IBM System/360 Models 30 and 40 stored the architected 32-bit registers in a magnetic core array known as local storage, while the integrated circuits implemented 8-bit and 16-bit microcode-accessible registers, respectively. See P. Fagg, J. Brown, J. Hipp, D. Doody, J. Fairclough, and J. Green, *IBM System 360 engineering*, Proc. AFIPS Fall Joint Computer Conf.

(FJCC), Pt. I, vol. 26, 1964, at 205-231. Therefore, when the Models 30 and 40 were running the System/360 microcode, the architected 32-bit registers were implemented in microcode (software) and not implemented in the hardware integrated circuits.

Floating Point Unit

16. Computers often perform operations with floating point numbers. The part of a computer system that performs operations on floating point numbers is typically called a "floating point unit."

17. Many of the floating point operations occur with numbers in "normalized" form, which is a representation of data that eliminates leading zeros (e.g., 2.2×10^{-2} instead of 0.022×10^0). A person of ordinary skill in the art at the time the '106 patent was filed would have understood that floating point operations on numbers in normalized form were typically performed in a hardware implementation of a floating point unit.

18. Other floating point operations occur with numbers in "denormalized" form, which are numbers that are so close to zero that if put in normalized form would have to be treated as zero due to limits in the floating point representations defined by the architecture (e.g., 0.02×10^{-200} cannot be represented in normalized form as 2.0×10^{-202} when -200 is the most negative exponent possible). A person of ordinary skill in the art at the time the '106 patent was filed would also have understood that floating point operations on numbers in denormalized form could have been performed by either hardware, software, or a combination of both. For example, the Intel i860 implemented floating point operations on numbers in denormalized form using software rather than hardware. See L. Kohn, and N. Margulis, *Introducing the Intel i860 64-bit Microprocessor*, IEEE Micro, vol. 9, no. 4, August 1994, at 15-30, available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=31485; J. Demmel and X. Li, *Faster*

Numerical Algorithms via Exception Handling, IEEE Transactions on Computers, vol. 43, no. 8, 1994, at. 983-992, available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=295860.

Host-Network Interface

19. A computer may be "partitioned." Each "partition" acts as a separate computer, runs its own operating system, and is completely isolated from the other partitions, even though the partitions may physically be part of the same computer.

20. A partitioned portion of a computer may connect to a network to communicate with other partitions and/or other computers. In 1998, at the time of the filing of the invention of the '812 patent, partitions of a computer often used a host-network interface to connect to a network.

21. Several IBM systems in 1998 used a host-network interface consisting of a host channel connection (e.g., ESCON) and an OSA adapter card. This host-network interface consisted of hardware and software (e.g., device drivers).

Firmware Image

22. Firmware is typically stored in a computer system on some form of non-volatile memory. For most computers, firmware is used to boot or "start-up" the computer.

23. In 2001, at the time of the inventions recited in the '002 patent, a person of ordinary skill in the art would have known that if a firmware image was copied by a partition from non-volatile memory, it could be stored in one of several locations, such as RAM. A person of ordinary skill in the art would also have known that the non-volatile memory storing the firmware image (before being copied) could have been one of many different storage types, such as an integrated circuit (i.e., chip), floppy disk, or CD; this non-volatile memory storage location was a design choice unrelated to the booting performed by the copy of the firmware

image. For example, the IBM System/370 Model 145 copied its firmware from a floppy disk into RAM memory as part of the boot process. *See* IBM SYSTEM/370 MODEL 145 FUNCTIONAL CHARACTERISTICS, IBM FORM NUMBER GA24-3557 (Oct. 1970), available at http://bitsavers.vt100.net/pdf/ibm/370/funcChar/GA24-3557-1_370-145_funcChar_Oct70.pdf.

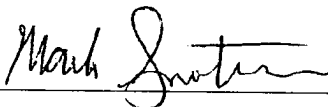
Routine

24. In 1997, one of ordinary skill in the art would have understood the term "routine" to mean a section of code that can be invoked by a program to perform a specific task. One purpose of a routine is to reduce the duplication of source code when writing the program. This allows different parts of the program to call the routine when necessary instead of rewriting the code for that task.

25. In general, the desired effect of a routine is the overall specific task that it performs rather than necessarily being a particular sequence of steps. For example, in a routine that adds three integers, a particular sequencing of the steps is not essential to performing the function; $1 + 2 + 3$ is the same as $2 + 3 + 1$. The sequence in which the instructions in a routine are executed may depend upon the data input. Routines typically include conditional logic, such as "if," "then," and "else" statements, and therefore the complete set of instructions that make up a routine may not all be executed for a given invocation of that routine. Furthermore, routines can call other routines either conditionally or unconditionally.

I declare under penalty of perjury that the forgoing is true and correct.

Executed at Clemson, SC, April 21, 2008



Mark Smotherman, Ph.D.

Mark Smotherman -- Curriculum Vitae

Department of Computer Science
209 McAdams Hall
Clemson University
Clemson, SC, 29634-0974

Email: mark@cs.clemson.edu
Office: (864) 656-5878
FAX: (864) 656-0145

EDUCATION

- Ph.D., Computer Science, University of North Carolina at Chapel Hill, August 1984.
 - Dissertation: Parametric Error Analysis and Coverage Approximations in Reliability Modeling (under the direction of Dr. Kishor Trivedi, Duke University)
- B.S., summa cum laude, Physics, Middle Tennessee State University, May 1977.

ACADEMIC EXPERIENCE

- Associate Professor, Computer Science, Clemson University, August 1990 to present.
- Director of Graduate Affairs, Computer Science, Clemson University, August 1991 to December 1992, and August 2004 to present.
- Assistant Professor, Computer Science, Clemson University, August 1984 to August 1990.

INDUSTRIAL EXPERIENCE

- Programmer, Systems Network Architecture Group, IBM Corp., Research Triangle Park, NC, Summer 1982.
- Programmer, Vanderbilt University Medical Center, Nashville, TN, Summer 1980.
- Programmer, IBM Cambridge Scientific Center, Cambridge, MA, Summer 1979.
- Analyst, Systems Programming, Bank of America Data Processing Systems Center, San Francisco, CA, Summer 1978.

CONSULTING AND REVIEWING

- Consultant, prior art in computer architecture and software.

- Expert witness (testifying), US District Court.
- Textbook reviewer, including detailed technical reviews of various editions of Hennessy and Patterson CA:AQA and Patterson and Hennessy COD:HSI for Morgan-Kaufmann.

PROFESSIONAL MEMBERSHIPS, HONORS, AND SERVICE

- ACM (Member)
- IEEE (Student Member 1980, Member 1984, Senior Member 1996)
- Space Act Award and Certificate from NASA Langley Research Center for HARP reliability prediction software and technical report (1997)
- IEEE Asia-Pacific Computer Systems Architecture Conference (ACSAC) 11/12/13, Program Committee Member (2006-2008)
- International Conference on Contemporary Computing (IC3), Program Committee Member (2008)
- Journal of Instruction-Level Parallelism, member of the founding editorial board (1997 to 2002)
- Journal of Parallel Processing, guest co-editor, special issue on Micro-30 (1999)
- ACM/IEEE International Symposium on Microarchitecture (MICRO) 31/32, Steering Committee Member (1998-1999)
- ACM/IEEE MICRO-30, General Chair (1997)
- ACM/IEEE MICRO-29, Program Committee Member (1996)

GRANTS

- Co-principal investigator in equipment grant for SUN HPC 3000 (M. Franklin, principal investigator, other co-PIs: D. Dawson, L. Thompson, and C. Rahn), NSF CISE, 1998.
- Co-principal investigator with M. Franklin, "Exploring Multiple Control Flows for ILP Processors," NSF CCR, August 1997, two years.
- Co-principal investigator in "Reconfigurable Computing Systems for Regional Validation Center Applications," (W. Ligon, principal investigator), NASA GSFC, August 1997, three years.
- Co-principal investigator in "An Application Framework for Reconfigurable Computing Systems," (W. Ligon, principal investigator), NASA GSFC, August 1997, three years.
- Participant in "AViiON Systems Performance" (R. Geist, principal investigator), Data General Corporation, August 1992, three years.

- Adviser for L. Winston, NSF CISE 1991 Minority Graduate Fellowship.
- Co-principal investigator with R. Geist, "Sufficiency and Necessity in Reliability Modeling," NASA LaRC, April 1987, thirty-nine months.
- Co-adviser with R. Geist for R. Talley, "Use of Two-Parameter Failure Distributions in Reliability Modeling," NASA Graduate Student Researchers Program, August 1987.
- Co-adviser with R. Geist for M. Brown, "HARP: Towards Phase III," NASA Graduate Student Researchers Program, August 1986.
- Co-principal investigator with R. Geist, "A Modeling Tool for Improved Product Design," Personal Computer Division, NCR Corporation, June 1986, one year.
- Co-principal investigator with R. Geist, "Cost Effectiveness in Solution Techniques for HARP Models," NASA LaRC through a subcontract with Duke University, October 1985, one year.
- Co-principal investigator with R. Geist, "Numerical Issues in the Design of the Hybrid Automated Reliability Predictor," NASA LaRC through a subcontract with Duke University, October 1984, one year.

PUBLICATIONS

COMPUTER ARCHITECTURE AND PERFORMANCE

- Book Chapters and Sections:
 - M. Smotherman, "Superscalar Processors," in V.G. Oklobdzija (ed.), CRC Computer Engineering Handbook, 2nd ed., CRC Press, Boca Raton, 2007, pp. 2-1 to 2-10. An earlier version appears in the 1st ed., 2002, pp. 6.1-6.6.
 - M. Smotherman, "Survey of Superscalar Processors," chapter 8 in J.P. Shen and M.H. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, McGraw-Hill, 2005, pp. 369-451.
 - M. Smotherman, "Microprogramming and Microarchitecture," in A. Kent and J. Williams (eds.), Encyclopedia of Microcomputers, volume 25, supplement 4, Marcel Dekker, New York, 2000, pp. 257-272. Also appears in A. Kent (ed.), Encyclopedia of Library and Information Science, volume 69, Marcel Dekker, New York, 2001, pp. 262-277.
 - M. Smotherman, "A Sequencing-Based Taxonomy of I/O Systems and Review of Historical Machines," Paper 1 of Chapter 7, in M. Hill, N. Jouppi, and G. Sohi (eds.), Readings in Computer Architecture, Morgan Kaufmann, San Francisco, 2000. Originally appeared in Computer Architecture News, vol. 17, no. 5, pp. 5-15, September 1989. (Special issue on input/output architecture)

- Journals:
 - M. Smotherman, M. Domeika, J. Watkins, and D. Suggs, "Instruction Cache Performance of a Commercial Workload on the Motorola 88110 Microprocessor," *Microprocessors and Microsystems*, vol. 20, no. 9, pp. 521-527, May 1997.
- Technical Conferences (fully refereed):
 - R. Geist, J. Hicks, M. Smotherman, and J. Westall, "Parallel Simulation of Petri Nets on Desktop PC Hardware," *Proceedings of the 2005 Winter Simulation Conference*, Orlando, FL, December 2005, pp. 374-383.
 - M. Smotherman and M. Franklin, "Improving CISC Instruction Decoding Performance Using a Fill Unit," *MICRO-28, 28th Annual International Symposium on Microarchitecture*, Ann Arbor, MI, November 1995, pp. 219-229.
 - M. Franklin and M. Smotherman, "A Fill-Unit Approach to Multiple Instruction Issue," *Proceedings of MICRO-27, 27th Annual International Symposium on Microarchitecture*, San Jose, CA, December 1994, pp. 162-171.
 - M. Smotherman, S. Chawla, J.S. Cox, and B. Malloy, "Instruction Scheduling for the Motorola 88110," *Proceedings of MICRO-26, 26th Annual International Symposium on Microarchitecture*, Austin, TX, December 1993, pp. 257-262.
 - M. Smotherman, S. Krishnamurthy, P. Aravind, and D. Hunnicutt, "Efficient DAG Construction and Heuristic Calculation for Instruction Scheduling," *Proceedings of MICRO-24, 24th Annual International Symposium on Microarchitecture*, Albuquerque, NM, November 1991, pp. 93-102.
- General Conferences:
 - M. Smotherman, "Understanding EPIC Architectures and Implementations," *40th Annual ACM Southeast Conference*, Raleigh, April 2002, pp. 71-78.
 - R. Geist, J. Westall, D. Tregliua, and M. Smotherman, "Real-time, 3-D Graphics for the Linux PC," *CMG98*, Anaheim, CA, December 1998.
 - S. Shirhatti and M. Smotherman, "A Technique for Obtaining Kernel Mode Address Traces on a Pentium-Based Linux System," *35th Annual ACM Southeast Conference*, Murfreesboro, TN, April 1997, pp. 57-59.
 - R. Geist, M. Smotherman, and M. Westall, "Performance Evaluation of NUMA Architectures," *34th Annual ACM Southeast Conference*, Tuskegee, AL, April 1996, pp. 78-85.
 - D. Bushey, A. Capes, A. Sankaran, and M. Smotherman, "Cache-Friendly FIFO Lists," *34th Annual ACM Southeast Conference*, Tuskegee, AL, April 1996, pp. 35-42.

- J. McCalpin and M. Smotherman, "Automatic Benchmark Generation for Cache Optimization of Matrix Operations," Proceedings 33rd Annual ACM Southeast Conference, Clemson, SC, March 1995, pp. 195-204.
- T.J. Tumlin and M. Smotherman, "An Evaluation of the Design of the Gamma 60," 3eme Colloque Histoire De L'Informatique, Sophia Antipolis, France, October 1993.
- Technical Reports:
 - M. Smotherman, "Preliminary Report on the Economics of Intelligent Terminals," Technical Report G320-2132, IBM Cambridge Scientific Center, Cambridge, MA, December 1980.

RELIABILITY MODELING

- Book Chapters:
 - M. Smotherman, "Transient Solution of Time-Inhomogeneous Markov Reward Models with Discontinuous Rates," in W. Stewart (ed.), Numerical Solution of Markov Chains, Marcel Dekker, New York, 1990.
- Journals:
 - M. Smotherman, W.T. Stinson, and M. Chetuperambal, "A Hybrid Model for Weekday Replacements with Infant Mortality Failures," Microelectronics and Reliability, vol. 38, no. 4, pp. 685-688, 1998.
 - M. Smotherman, "Error Analysis in Analytic Reliability Modeling," Microelectronics and Reliability, vol. 30, no. 1, pp. 141-149, 1990.
 - M. Smotherman and R. Geist, "Phased Mission Effectiveness Using a Nonhomogeneous Markov Reward Model," Reliability Engineering and System Safety, vol. 27, no. 2, pp. 241-255, 1990.
 - M. Smotherman and K. Zemoudeh, "A Non-Homogeneous Markov Model for Phased-Mission Reliability Analysis," IEEE Transactions on Reliability, vol. 38, no. 5, pp. 585-590, December 1989.
 - R. Geist, M. Smotherman, K. Trivedi, and J. Dugan, "The Use of Weibull Fault Processes in Modeling Fault Tolerant Systems," AIAA Journal of Guidance, Control, and Dynamics, vol. 11, no. 1, pp. 91-93, January-February 1988. (Engineering Notes section)
 - R. Geist, M. Smotherman, K. Trivedi, and J. Dugan, "The Reliability of Life-Critical Computer Systems," Acta Informatica, vol. 23, no. 6., pp. 621-642, November 1986.
 - J. Dugan, K. Trivedi, M. Smotherman, and R. Geist, "The Hybrid Automated Reliability Predictor," AIAA Journal of Guidance, Control, and Dynamics, vol. 9,

no. 3, pp. 319-331, May-June 1986.

- M. Smotherman, R. Geist, and K. Trivedi, "Provably Conservative Approximations to Complex Reliability Models," IEEE Transactions on Computers, vol. C-35, no. 4, pp. 333-338, April 1986. (Special issue on fault-tolerant computing)
- J. McGough, M. Smotherman, and K. Trivedi, "The Conservativeness of Reliability Estimates Based on Instantaneous Coverage," IEEE Transactions on Computers, vol. C-34, no. 7, pp. 602-609, July 1985. (Reviewed in ACM Computing Reviews 8606-0508)
- K. Trivedi, J. Dugan, R. Geist, and M. Smotherman, "Hybrid Reliability Modeling of Fault-Tolerant Computer Systems," Computers and Electrical Engineering, vol. 11, nos. 2+3, pp. 87-108, 1984. (Special issue on reliability and verification of computing systems, reviewed in ACM Computing Reviews 8605-0415)
- Technical Conferences (fully refereed):
 - R. Geist, M. Smotherman, and R. Talley, "Modeling Recovery Time Distributions in Ultrareliable Fault Tolerant Systems," Proceedings 20th International Symposium on Fault-Tolerant Computing, Newcastle upon Tyne, UK, June 1990, pp. 499-504.
 - R. Geist, M. Smotherman, and M. Brown, "Ultrahigh Reliability Estimates for Systems Exhibiting Globally Time-Dependent Failure Processes," Proceedings 19th International Symposium on Fault-Tolerant Computing, Chicago, IL, June 1989, pp. 152-158.
 - R. Geist and M. Smotherman, "Ultrahigh Reliability Estimates Through Simulation," Proceedings Annual Reliability and Maintainability Symposium, Atlanta, GA, January 1989, pp. 350-355.
 - K. Trivedi, J. Dugan, R. Geist, and M. Smotherman, "Issues in Reliability Modeling of Fault-Tolerant Computers," Proceedings Second GI/NTG/GMR Conference on Fault-Tolerant Computing Systems, Bonn, West Germany, September 1984, pp. 228-239.
 - K. Trivedi, J. Dugan, R. Geist, and M. Smotherman, "Modeling Imperfect Coverage in Fault-Tolerant Systems," Proceedings 14th International Symposium on Fault-Tolerant Computing, Orlando, FL, June 1984, pp. 77-82.
 - R. Geist, K. Trivedi, J. Dugan, and M. Smotherman, "The Design of the Hybrid Automated Reliability Predictor," Proceedings IEEE/AIAA Fifth Digital Avionics Systems Conference, Seattle, WA, November 1983, pp. 16.5.1-16.5.8.
- Technical Reports:
 - S. Bavuso, E. Rothmann, J. Bechta Dugan, K. Trivedi, N. Mittal, M. Boyd, R.

Geist, and M. Smotherman, "HiRel: Hybrid Automated Reliability Predictor (HARP) Integrated Reliability Tool System (Version 7.0)," NASA Technical Paper 3452, Volume 1, NASA Langley Research Center, Hampton, VA, November 1994.

- M. Smotherman, "Phased Mission Analysis Using Nonhomogeneous Markov Models," 1991 System Evaluation and Assessment Technology Workshop, Naval Surface Warfare Center, Silver Spring, MD, August 1991.
- K. Trivedi, J. Dugan, R. Geist, and M. Smotherman, "HARP: Theory, Implementation and Applications," Technical Report CS-1987-2, Dept. of Computer Science, Duke University, Durham, NC, January 1987.

COMPUTER SCIENCE EDUCATION

- General Conferences (refereed):
 - M. Smotherman, "Examining Compiled Code," Proceedings 20th Annual ACM SIGCSE Technical Symposium, Louisville, KY, February 1989, pp. 165-169.
 - R. Little and M. Smotherman, "Assembly Language Courses in Transition," Proceedings 19th Annual ACM SIGCSE Technical Symposium, Atlanta, GA, February 1988, pp. 95-99.
 - M. Smotherman, "On the Use of Naming and Binding in Early Courses," Proceedings 18th Annual ACM SIGCSE Technical Symposium, St. Louis, MO, February 1987, pp. 79-83.

PROGRAMMING LANGUAGES AND OPERATING SYSTEMS

- Technical Conferences (fully refereed):
 - M. Jazayeri, C. Ghezzi, D. Hoffman, D. Middleton, and M. Smotherman, "CSP/80: A Language for Communicating Sequential Processes," Proceedings IEEE COMPCON, Washington, DC, September 1980, pp. 736-740.
 - M. Jazayeri, C. Ghezzi, D. Hoffman, D. Middleton, and M. Smotherman, "Design and Implementation of a Language for Communicating Sequential Processes," Proceedings International Conference on Parallel Processing, Harbor Springs, MI, August 1980, pp. 173-180.
- General Conferences:
 - J. Westall and M. Smotherman, "Dynamic Control of Job Arrival Distributions," Proceedings 24th Annual ACM Southeast Regional Conference, Tampa, FL, April 1986, pp. 156-162.
- Technical Reports:
 - W. Madison and M. Smotherman, "Guarded Remote Procedure Calls As

Synchronization Constructs for C," Technical Report 91-102, Dept. of Computer Science, Clemson University, February 1991.

STUDENT SUPERVISION

- Mark Harris, M.S., December 1986, "Extending Microcode Compaction for Real Architectures" (scholarly paper), appears in Proceedings 20th Annual Workshop on Microprogramming, Colorado Springs, December 1987, pp. 40-53.
- Guillermo Roa, M.S., December 1987, "An Analytic Model for the Performance Prediction of Modern Microcomputers" (scholarly paper), part of which appears in G. Roa and C. Reynolds, "A Tool for the Design of Personal Computing Systems," Proceedings 25th Annual ACM Southeast Regional Conference, April 1987, pp. 535-539.
- Vasudevan Subramanian, M.S., December 1987, "Performance Prediction of Unix Systems Using A Linear Regression Model" (scholarly paper).
- Robert Schwartz, M.S., August 1989, "The Design and Development of a Dynamic Program Behavior Measurement Tool for the Intel 8086/88" (scholarly paper), appears in Computer Architecture News, vol. 17, no. 4, pp. 82-94, June 1989.
- Jim O'Connor, M.S., December 1989, "Error Analysis in Highly Reliable Systems" (scholarly paper), appears in Proceedings 27th Annual ACM Southeast Regional Conference, Atlanta, April 1989, pp. 428-431.
- Steve O'Neal, M.S., December 1989, "A SPARC Implementation Simulator for Code Optimization Evaluation" (scholarly paper).
- Sanjay Krishnamurthy, M.S., May 1990, "Static Scheduling of Multi-Cycle Operations for a RISC Processor" (scholarly paper), part of which was reported at a poster session at Supercomputing '90 and part of which appears as "A Brief Survey of Papers on Scheduling for Pipelined Processors," SIGPLAN Notices, vol. 25, no. 7, pp. 97-106, July 1990 (see also MICRO-24 joint paper).
- David Hunnicutt, M.S., May 1991, "DDG Construction Algorithms" (scholarly paper) (see also MICRO-24 joint paper).
- Stig Thormodsrud, M.S., August 1991, "Post-Link Instruction Scheduling" (scholarly paper).
- Lokender Bommisetty, M.S., December 1991, "A Study of Non-DAG Based Instruction Scheduling Techniques" (scholarly paper).
- P.S. Aravind, M.S., May 1992, "Selectable Heuristics for Instruction Scheduling" (scholarly paper) (see also MICRO-24 joint paper).
- Stan Cox, M.S., December 1992, "Code Scheduling for the MC88110 Superscalar

RISC Processor Using Reservation Tables" (scholarly paper), part of which appears in MICRO-26 joint paper.

- Keerti Rane, M.S., August 1993, "Effects of Software Prefetching and Tiling on the Performance of Matrix Multiplication for the 88110 Superscalar RISC Processor" (scholarly paper).
- Shuchi Chawla, M.S., December 1993, "Instruction Scheduling for the Motorola 88110 Superscalar Processor" (scholarly paper), part of which appears in MICRO-26 joint paper.
- Jasbir Manotra, M.S., December 1994, "Algorithm Execution Animation for Matrix Multiply" (scholarly paper).
- Mazin Ramadan, M.S., December 1995, "GURU: A Retargetable CFG-Based Program Reorganizer" (scholarly paper), part of which appears in Proceedings 34th Annual ACM Southeast Regional Conference, Tuskegee AL, April 1996, pp. 164-169 (with joint author Devidas Gupta).
- Kishore Sirivelu, M.S., May 1996, "A Survey of Basic Block Reordering Algorithms" (scholarly paper), appears in Proceedings 34th Annual ACM Southeast Regional Conference, Tuskegee AL, April 1996, pp. 182-187.
- Jane Watkins, M.S., May 1996, "Analysis of Level-One Instruction Cache Designs For A Commercial Workload" (scholarly paper), part of which appears as "Instruction Cache Miss Ratio Analysis of a Commercial Workload on the Motorola 88110," Proceedings 34th Annual ACM Southeast Regional Conference, Tuskegee AL, April 1996, pp. 260-264.
- Bharath Chandramohan, M.S., December 1996, "Hardware Prefetching Techniques for List Structures" (scholarly paper).
- Sachin Shirhatti, M.S., May 1997, "A Technique for Observing Kernel Mode Activity on a Pentium-Based Linux System" (scholarly paper), part of which appears in Proceedings 35th Annual ACM Southeast Conference, Murfreesboro, TN, April 1997, pp. 57-59.
- Geetha Ravishankar, M.S., August 1997, "Study of Call Behavior of IBS Traces" (scholarly paper).
- Bun Kea Tan, M.S., May 1998, "Optimizing the Linux Kernel" (scholarly paper).
- Xinlei Wu, M.S., May 1999, "Study of Call Behavior of Desktop Applications on Windows NT" (scholarly paper).
- Natasha Pothen, M.S., May 1999, "Performance Considerations for Multithreaded Programs on Small Scale Shared Memory Multiprocessors" (scholarly paper).
- Chris Freeze, M.S., August 2000, "Performance Measuring of Modern Multiuser Operating Systems" (scholarly paper).

- Kartik Agarwal, M.S., May 2001, presentation of T. Mitchem, R. Lu, and R. O'Brien, "Using Kernel Hypervisors to Secure Applications" (research experience).
- Malathi Molugu, M.S., May 2003, presentation of R. Cooksey, S. Jourdan, and D. Grunwald, "A Stateless, Content-Directed Data Prefetching Mechanism" (research experience).
- Sharath Rao, M.S., December 2003, presentation of D. Marr, et al., "Hyper-Threading Technology: Architecture and Microarchitecture" (research experience).
- Prachi Bagayatkhar, M.S., December 2003, presentation of T.-C. Chiueh and P. Pradhan, "Cache Memory Design for Network Processors" (research experience).
- Mark Bidewell, M.S., expected May 2005, "Software-Based Dynamic Thermal Management for Linux Systems" (thesis).

INSTRUCTIONAL ACTIVITIES AT CLEMSON

- Undergraduate Courses Taught
 - CPSC 102 Computer Science II (Spring '95)
 - CPSC 157 Introduction to C Programming (Summer '97)
 - CPSC 215 Tools and Techniques for Software Development (Spring '01, Summer '03 '05 '06)
 - CPSC 230 Assembly Language Programming (Spring '87 '88, Summer '88, Fall '88)
 - CPSC 231 Introduction to Computer Organization (Spring '97 '99 '00 '01 '05 '06 '07, Summer '92 '93 '94 '95 '96 '97 '99 '00 '01, Fall '96 '99 '00 '01 '04 '05 '06 '07)
 - CPSC 330 Computer Systems Organization (Spring '03 '04 '05 '06 '07 '08, Summer '91, Fall '02 '03 '04 '05 '06 '07)
- Senior/Graduate Courses Taught
 - CPSC 422/622 Introduction to Operating Systems (Spring '86 '88 '90 '97 '98 '99 '00 '02, Summer '87 '98 '99 '00 '01 '04, Fall '86 '87 '89 '94)
 - CPSC 423/623 Operating Systems Implementation (Spring '89 '90 '93, Fall '85 '88 '90 '92 '93)
 - CPSC 430/630 Computer Performance Evaluation (Spring '85 '86)
 - CPSC 435/635 Microprogramming (Fall '84 '85)
 - CPSC 464/664 Introduction to Computer Architecture (Spring '91 '95 '96 '02 '03 '04, Fall '86 '87 '89 '90 '91 '95 '96 '97 '98 '99 '00 '01 '02 '03)
 - CPSC 495 Honors Thesis Research (Spring '97)
- Graduate-Only Courses Taught
 - CPSC 830 Systems Modeling (Spring '94 '96 '98)
 - CPSC 864 Computer Architecture (Spring '89 '91, Fall '91 '92 '93 '94 '95 '96 '97 '98)
 - Special Topics courses in performance analysis, reliability modeling, and fault-tolerant computing (Spring '87, Summer '86 '88 '89 '90)
 - Seminars in computer architecture